Skoltech ML projects 2023

February 14, 2023

Daniil Selikhanovych, Telegram - @Dan_Sel

Warning : all projects are connected with deep learning for computer vision, so the team must be sure that it has GPUs with a capacity of at least 12 GBs. In case you have GPUs only on Google Colab without Premium, we suggest you to save models/optimizers on your Google Disk during the training since it may be interrupted.

Project 1: Novel view synthesis with neural radiance fields

Paper: `https://arxiv.org/pdf/2003.08934.pdf`
Implementation: `https://github.com/yenchenlin/nerf-pytorch`, `https://github.com/colmap/colmap`
Keywords: 3D reconstruction, volume rendering, MLP, neural rendering, rendering equation, Structure-from-Motion, Multi-View Stereo.

Neural Radiance Field (NeRF) models are novel view synthesis methods which use volume rendering with implicit neural scene representation via Multi Layer Perceptrons (MLPs). First introduced in ECCV 2020 by Mildenhall et al. (link), NeRF has achieved state of the art visual quality, produced impressive demonstrations, and inspired many subsequent works derived from this novel method. In the recent past, NeRF models have found applications in photo-editing, 3D surface extraction, and large/city-scale 3D representation and view synthesis.

In this project you will reproduce some results of NeRF algorithm for the problem of novel view synthesis. You will compare two scenarios for this problem: synthetic and real. In the synthetic case both train and test images have exact camera-to-world matrices and intrinsic parameters like focal length and principal point coordinates. In the real case only images of the scene are available and you have to extract estimation of camera-to-world matrices and intrinsic parameters by COLMAP framework, which implements Structure-from-Motion and Multi-View Stereo methods.

Replication tasks :

1. Derive and explain the formula for casting rays in the code (link). For this task plot coordinate system used in NeRF convention and use similar triangles.

2. Download synthetic Lego dataset and real LLFF Fern dataset in the NeRF paper. Train NeRF model on these datasets, report mean PSNR and SSIM metrics on train

and test images. Report mean reconstruction time per 1 image and training curves. Do your novel images look realistic? Which artifacts can you observe?

3. Extract train and test camera-to-world matrices and intrinsic parameters by COLMAP framework. It is possible that COLMAP will not converge, so try to change the parameters or restart it. Retrain NeRF on the COLMAP poses and compare results with the case when poses are known. Is the difference between these two setups significant in terms of PSNR and SSIM metrics?

Research tasks :

1. You can see that NeRF training for only one scene may need > 8 hours GPU training and rendering 1 image may need 30-40 seconds. The reason is due to many calls of MLP for rendering 1 image: for image with size 800×800 with 200 sampled points per ray we have to call MLP $800 \times 800 \times 200 = 128M$ times. Many works tried to improve training and inference time based on volume grids or training time based on hash. Two teams may try to explore one of these research directions:

   (a) The first team can try to test methods based on grids like Direct Voxel Grid optimization (paper - link, code -link). The task is to understand the theory behind these methods and compare results with vanilla NeRF from the previous part. Other papers should be discussed with the author of the project.

   (b) The second team can try to test methods based on hash idea like Instant-NGP (paper - link, code - link). The task is to understand the theory behind these methods and compare results with vanilla NeRF from the previous part. Other papers should be discussed with the author of the project.

Project 2: Spatially-Adaptive Pixelwise Networks for Fast Image Translation

Paper: `https://arxiv.org/abs/2012.02992`
Implementation: `https://github.com/tamarott/ASAPNet`
Keywords: image-to-image translation, segmentation, multi-layer perceptron, pix2pix, denoising

In this paper, the authors propose an alternative to the standard approach for image-to-image task, which is based on generative adversarial networks. The first idea is to use pixel-wise MLPs during processing high-resolution image instead of convolution neural network, so each pixel is processed independently of others. Secondly, these parameters of MLPs are predicted by a fast convolutional network that processes an low-resolution representation of the input. According to authors, such model (ASAPNet) is up to 18x faster than state-of-the-art baselines and gives comparable results.

In this project you will reproduce some key results of this paper for segmentation problem. Moreover, you have to research the possibility of the usage such network for other image-to-image translation task: denoising problem.

Replication tasks :

1. Download and prepare Cityscapes 256×512 dataset. Prepare generator and discriminator networks for training with parameters that are described in the paper.

2. Train proposed model for segmentation task on prepared training dataset from scratch. Compute Frechet Inception Distance (FID) and mean Intersection-over-Union (IoU) for the validation dataset. Compare results with values obtained by the authors in the paper. Report inference time and specification for the used GPUs.

3. Train pix2pix model for the same data and report the same results as for the model of authors. Is ASAPNet faster than pix2pix model?

Research tasks :

1. In this part of the project you have to try to use such generator network for non-blind denoising. Download clean real images x from Berkeley Segmentation Dataset (BSD500). Split data into train and validation parts - 432 images for training and 68 images for validation.

2. For the training data create a synthetic noisy images $y = x + \sigma \cdot n, n \sim \mathcal{N}(0, I)$ with the different standard deviations σ of noise level: from 10 to 55 (for the images scaled from 0 to 255 values in the pixels). For the validation data create a synthetic noisy images with the following standard deviations σ of noise level: 15, 25 and 50. Train any fast enough CNN-based non-blind denoising baseline for prepared data. For example, consider DnCNN model from this paper: `https://arxiv.org/abs/1608.03981` (code: `https://github.com/SaoYan/DnCNN-PyTorch`). Report PSNR and SSIM metrics for the results on the validation data and inference time as well.

3. Try to use generator architecture of ASAPNet from segmentation task for the same denoising problem. Compare results with baseline from the both views: PSNR/SSIM metrics and inference time. Was it good idea to use pixelwise network for this task?

Project 3: Vision Transformers for Image Restoration Problems

Relevant papers: `https://arxiv.org/abs/2108.10257`, `https://arxiv.org/abs/1711.07807`, `https://www.ceremade.dauphine.fr/~carlier/FISTA`.
Implementation: `https://github.com/JingyunLiang/SwinIR`
Keywords: vision transformers, attention mechanism, image restoration problems, ISTA, FISTA, denoising, deblurring.

Warning : Vision Transformers (ViT) are very computationally expensive models, so the team must be sure that it has several GPUs and distributed learning is possible.

Recently vision Transformer has shown great promise as it integrates the advantages of both CNN and Transformer. On the one hand, it has the advantage of CNN to process image with large size due to the local attention mechanism. On the other hand, it has the advantage of Transformer to model long-range dependency with the shifted window scheme. In this paper authors suggested how to adapt ViT for image restoration problems like denoising and super-resolution.

In this project you will reproduce results of this paper for denoising problem. After that you have to research the possibility of the usage such denoising model for other image-to-image translation task based on the ISTA and FISTA optimization schemes: non-blind image deblurring.

Replication tasks :

1. Download clean real images x from Berkeley Segmentation Dataset (BSD500). Split data into train and validation parts - 432 images for training and 68 images for validation.

2. For the training data create a synthetic noisy images $y = x + \sigma \cdot n, n \sim \mathcal{N}(0, I)$ with the different standard deviations σ of noise level: from 10 to 55 (for the images scaled from 0 to 255 values in the pixels). For the validation data create a synthetic noisy images with the following standard deviations σ of noise level: 15, 25 and 50. Train proposed SwinIR model for denoising task on prepared training dataset from scratch. Compute PSNR and SSIM metrics for the validation dataset. Compare results with values obtained by the authors in the paper. Report inference time and specification for the used GPUs.

3. SwinIR model according to the authors code is dealing with blind denoising. It means that the model doesn't need any information about noise level σ during the inference, so only noisy image y is enough. Try to add projection layer from the paper `https://arxiv.org/abs/1711.07807` to convert the model dealing with blind denoising input y into non-blind model dealing with input (y, σ). Does prior information of the noise level σ of validation data increase PSNR and SSIM metrics?

Research tasks :

1. In this part of the project you are supposed to develop the algorithm that can use trained denoising models for the other image restoration problem: image deblurring. Non-blind image deblurring is the task of restoration clean image x given blurred observation y, degradation operator H that represents convolution with known kernel k and standard deviation of the noise level σ. So, we consider the following linear degradation process:

$$y = Hx + \sigma \cdot n, \quad n \sim \mathcal{N}(0, I),$$

where $y \in \mathbb{R}^n$ and $x \in \mathbb{R}^m$ are vector representations of blurred and clean images respectively. $H \in \mathbb{R}^{n \times m}$ matrix represents valid convolution of clean image x with known kernel k. Read the paper `https://www.ceremade.dauphine.fr/~carlier/FISTA` about ISTA and FISTA optimization algorithms. Suppose we can solve denoising problem with the given level of noise σ:

$$T_{\sigma^2}(y) \stackrel{\text{def}}{=} \arg\min_u \left\{ g(u) + \frac{1}{2\sigma^2} \|u - y\|_2^2 \right\}.$$

Derive formulas based on ISTA and FISTA algorithms that solve the deblurring problem

$$T_{H,\sigma^2}(y) \stackrel{\text{def}}{=} \arg\min_u \left\{ g(u) + \frac{1}{2\sigma^2} \|Hu - y\|_2^2 \right\}$$

2. Download kernels k from the Sun deblurring dataset (`https://cs.brown.edu/people/lbsun/deblur2013/deblur2013iccp.html`) and ground truth images x (`http://cs.brown.edu/~lbsun/SRproj2012/Sun_Hays_SR_groundtruth.zip`). Prepare synthetic blurred images using the degradation model

$$y = k \circledast x + \sigma \cdot n, \quad n \sim \mathcal{N}(0, I).$$

Use noise levels $\sigma = 2.55, 12.75$.

3. Implement ISTA and FISTA algorithms with trained SwinIR as denoising model for deblurring of the blurred Sun images. Use any simple initialization for x_0 like y or results of the Wiener filter (for example, this implementation `https://scikit-image.org/docs/dev/api/skimage.restoration.html#skimage.restoration.wiener`). Use some padding for the H operator to be sure that x and y have the same sizes. Plot results of PSNR and SSIM metrics for different number of ISTA/FISTA steps. Report inference time and specification for the used GPUs. Does this algorithm perform well compared with deblurring baselines reported for the Sun dataset in this paper link? Due to the fact that model is not tuned specifically for the deblurring task, some artifacts around the borderds may appear, so it is better to compute metrics after cropping some pixels from the borders (for example, 50 from the each side).

Project 4: Non-uniform Blur Kernel Estimation via Adaptive Basis Decomposition

Relevant papers: `https://arxiv.org/abs/2102.01026`, `https://arxiv.org/pdf/1711.07064.pdf`.
Code: `https://github.com/GuillermoCarbajal/NonUniformBlurKernelEstimation`, `https://github.com/KupynOrest/DeblurGAN`.
Keywords: blind deblurring, non-uniform deblurring, total variation, MAP estimation, Richardson-Lucy iteration, GAN for deblurring.

Warning : No open code for the training and data preparation, only inference pretrained model! So no one guarantees that the results will be reproducible.

In this paper, the authors propose the model of non-uniform motion blur, which generalize the following common used uniform model of blurring process:

$$y = Hx + \sigma \cdot n, \quad n \sim \mathcal{N}(0, I).$$

where $y \in \mathbb{R}^n$ and $x \in \mathbb{R}^m$ are vector representations of blurred and clean images respectively. $H \in \mathbb{R}^{n \times m}$ matrix represents valid convolution of clean image x with some kernel k. The key idea is to use not one kernel for the whole image, but per-pixel blur kernels k_i. Such idea applied to real motion-blurred images gives impressive reported results, but the authors didn't release the code of the model training and data preparation.

In this project you are supposed to code and reproduce the results of the authors for Kohler and Gopro datasets with real blur.

Replication tasks :

1. Download pretrained model from the authors repo and make inference on the Kohler (`https://webdav.tuebingen.mpg.de/pixel/benchmark4camerashake/`) and Gopro (`https://seungjunnah.github.io/Datasets/gopro.html`) datasets. Compute PSNR and SSIM metrics. Compare them with results reported in the paper. Are they reproducible?

   Warning : for Kohler dataset you may deal with Matlab code for PSNR computation. You can start from Gopro dataset.

2. Try test pretrained DeblurGAN model from the repo `https://github.com/KupynOrest/DeblurGAN` on the same data - Kohler and Gopro datasets. Compute PSNR and SSIM metrics. Compare them with results reported in the paper `https://arxiv.org/abs/2102.01026`. Are they reproducible? If results are worse you may try to retrain DeblurGAN based on the work `https://arxiv.org/pdf/1711.07064.pdf` and its code from the repo `https://arxiv.org/pdf/1711.07064.pdf`.

Research tasks :

1. Try to write a code for the training of model from the paper `https://arxiv.org/pdf/2102.01026.pdf`. You may use any open datasets for training. Plot training and validation metrics dependence on the number of epoch. Are results from the paper reproducible?

2. In case of failure to reproduce the results from the paper `https://arxiv.org/abs/2102.01026` try to investigate the paper about DeblurGAN. For this case your task is to reproduce Table 1 and Table 2 from `https://arxiv.org/abs/2102.01026` and train the networks from scratch.

Project 5: Stochastic Differential Equations for Generative Modeling

Relevant papers: `https://openreview.net/pdf?id=PxTIG12RRHS`, `https://arxiv.org/abs/2006.11239`, `https://arxiv.org/abs/2003.06060`.
Code: `https://github.com/yang-song/score_sde_pytorch`, `https://github.com/lucidrains/denoising-diffusion-pytorch`.
Keywords: generative modeling, stochastic differential equations, diffusion models, Langevin dynamics, Markov chains, neural ODEs, GANs, MCMC.

Warning : Image generation with denoising diffusion probabilistic modeling may be not fast, so it is better to start experiments as soon as possible. According to repo, PyTorch model consumes 20.6 GB in total, so you may need more than 1 GPU. You can also try to reduce the size of the model and decrease batch size according to available computational resources.

Score-based generative models have proven effective at generation of images and audio. Moreover, recent results shows that these model outperform state-of-the-art generative adversarial networks in the generative modeling problems. Such model construct a diffusion process $\{x(t)\}_{t=0}^{T}$ indexed by a continuous time variable $t \in [0, T]$ such that $x(0) \sim p_0$, for which we have a dataset of i.i.d. samples, and $x(T) \sim p_T$, for which we have a tractable form to generate samples efficiently. This diffusion process can be modeled as the solution to an Ito stochastic differential equation. There are some different discretization strategies for SDE like Score matching with Langevin dynamics (SMLD) and Denoising diffusion probabilistic modeling (DDPM).

In this project you are supposed to compare different SDE solvers for generative modeling problem and reproduce some results of the paper `https://openreview.net/pdf?id=PxTIG12RRHS` for CIFAR10 and/or Celeba datasets.

Replication tasks :

1. Compare different predictor-corrector samplers for CIFAT10 dataset: try to reproduce Table 1 from `https://openreview.net/pdf?id=PxTIG12RRHS` for ancestral

sampling and reverse diffusion. Report best achieved Frechet Inception Score (FID) and training/inference time. Plot the dynamics of FID for different numbers of diffusion steps N. Does combination of corrector and predictor steps really helps compared with only predictor-only samplers and corrector-only samplers?

2. The same experiment, but for Celeba dataset scaled to the size 64×64. Compare results with Figure 10. Try to study different architectures like in the Figure 10. For example, try different numbers (2 and 4) and types of residual blocks (from DDPM and from BigGAN).

Research tasks :

1. In this part of the project you are supposed to study GANs as energy based models (EBM-GAN) from this paper: `https://arxiv.org/abs/2003.06060`. Key idea is to use Langevin dynamics for the latent variables of the generator to correct samples of the given model. So here we also are interested in the sampling process like in the diffusion models. For this problem you should train GANs with comparible architectures or download some pretrained models for CIFAR10. You will need ***both*** generator and discriminator. Theory of the EBM-GAN states that the models have to be trained for the standard GAN loss $\mathbb{E}_{x\sim p_d(x)}[\log D(x)]+\mathbb{E}_{z\sim p(z)}[\log(1-D(G(z)))]$ or WGAN loss $\mathbb{E}_{x\sim p_d(x)}D(x) - \mathbb{E}_{z\sim p(z)}D(G(z))$. Then code the Langevin dynamics in the latent space as it is described in the paper `https://arxiv.org/abs/2003.06060` - see Algorithm 1 in the paper. Plot the dynamics of FID for different numbers of Langevin steps N. You may try to vary not only step size ε, but standard deviation of the noise σ as well (theory states that $\sigma = \sqrt{2\varepsilon}$ is a good choice, but in practice varying of σ also can be useful). Does MCMC in the latent space work for your model? Report the time for MCMC sampling process.

2. Compare the results of DDPM/SMLD experiments with EBM-GAN. For the fair comparison fix time of the computations, so running diffusion process for DDPM/SMLD will take approximately the same time as Langevin dynamics for EBM-GAN. Which model performs better for the fixed time?